System on Chip Approach for Onboard Computer

Shubham Agarwal[#], Rajiv Bhatia, N Padmavathi, S Sudhakar, S Udupa
Control Electronics Group, ISRO Satellite Centre
Indian Space Research Organization
Bangalore, India
[#]shubham@isac.gov.in

***Abstract*—This paper deals with the design and development of a System on chip [SoC] based Onboard Computer [OBC] for future onboard space applications of Indian Space Research Organization [ISRO]. The System on Chip approach shall integrate processor core with associated peripherals, other standard cores like MIL-STD-1553B core, application specific low power digital and analog circuits on a monolithic mixed-signal radiation hardened Application Specific Integrated Circuit [ASIC]. The open source LEON3 Processor core has been chosen for the central processing unit of the SoC after a detailed comparative study. The LEON3 processor has been configured modified, integrated with in-house designed OBC logics and implemented on a Xilinx Virtex 5 FPGA. The SoC can easily interface to other satellite subsystems since a interface for the Mil-Std 1553 Bus has also been designed and implemented. A Floating Point unit has been integrated with the LEON3 pipeline to accelerate the computation of complex floating point algorithms. A closed loop design of the onboard software was executed on the LEON3 processor to obtain a performance gain of nearly 50 when compared to presently used MAR31750 processor.**

Keywords— Field Programmable Gate Array, Onboard Computer, Processor, System on Chip, Clock Domain Crossing, LEON3, VLSI, Space avionics.

There is a growing need in Space industry for a high performance processor which can be used for a very long time without any embargo and which provides additional features like scalability, reconfigurability etc. Although the presently used MA31750 processor has provided ISRO with performance of upto 1 Million Instructions per second [MIPS], there is an ever increasing demand for an even more powerful microprocessor that will cater to the needs of future missions like Cartosat3, Geosat Missions. It is anticipated [1] that the future space missions shall require:

- At least 50 MIPS/25MFLOPS of performance
- Progressive interfaces like SpaceWire, CAN bus, MIL-STD-1553B bus, etc
- Unrestricted long term availability of components and software tools
- Reconfigurability and scalability
- Increased/inbuilt Fault tolerance

Based on the above requirements, a Technology Development Project [TDP] has been initiated aiming at design and development of OBC with such a processor. In order to utilize the advantages of the recent advances in Very Large Scale Integration [VLSI], it has been decided to implement the OBC as SoC [2] The SoC concept, in due course, shall integrate the all the digital logics of OBC, standard IP cores such as MIL-STD-1553B, RAM, etc in an ASIC. The SoC approach has been chosen since it shall help meet the following objectives:

- Reduce Mass ,Volume and Power requirements
- Improve Performance and Capability
- Improve reliability and fault tolerance
- Improve testability

A conceptual design of the SoC OBC is shown in fig. 1. The SoC TDP has been divided into two phases. The first phase shall see the design of the digital logics of SoC while the second phase will comprise of the analog circuits design activities. The first activity of the TDP consisted of the selection of processor core, which shall serve as Central Processing Unit [CPU] of OBC. A mix of hard cores and soft cores processor was surveyed thoroughly [3]-[7]. The results of the processor comparison are summarized in Table I.

TABLE I. PROCESSOR (SPACE GRADE) COMPARISON CHART

Processor	***LEON3***	***RAD750***	***Mongoose***	***OR1200***
Platform	FPGA, ASIC	ASIC	ASIC	FPGA, ASIC
Pipeline	7	6	5	5
DMIPS	175[FPGA]	>266	~12	20 @ 40MHz
Gate Count	< 25 k	~135k	>150k	~25k
MMU	Shared	Split	Split	Split
License Cost	Low	High	High	Low
Reconfigurability	Yes	No	No	Yes
Fault Tolerance	ECC and TMR	ECC	ECC	ECC

The LEON3 soft CPU core was selected as provides better performance, has a Fault Tolerant version and promises long term non-proprietary steady supply of processor chips unlike many other processors whose supplies are subjected to changing foreign export regulations [8].

978-1-4673-6190-3/13/$31.00 ©2013 IEEE

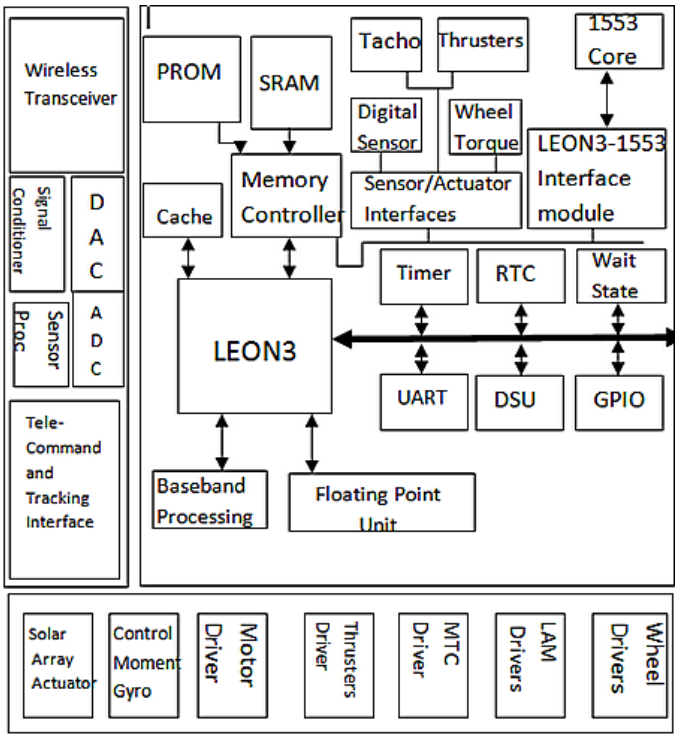

Fig. 1. Architecture of System-on-Chip for future ISRO Satellites

LEON3 is an open-source VHDL model of a 32-bit processing core that is fully compliant with the standard IEEE-1754 SPARC V8 architecture. The core comes with SPARC V8 compliant integer unit complete with hardware multiply, divide and MAC units. LEON3 has a pipeline depth of 7 stages [9]-[11]. LEON3 core features Harvard memory architecture, and a configurable set-associative cache subsystem. The number of registers in its register file is configurable within the SPARC V8 standard (2 to 32 registers). LEON3 also provides an interface to one of several available floating point units (FPU) cores as well as custom co-processors. It also includes support for an optional debug unit, timers, watchdogs, UARTs and interrupt controller.

The VHDL model of LEON3 is built around open source IP cores. These IP cores were integrated with the processor core in a hierarchal manner.

I. DESIGN AND SIMULATION

The design of LEON3 based SoC is dictated by the requirements of the OBC. The OBC interfaces are both logic and I/O intensive. The design of SoC includes interfacing of the OBC logics to the processor bus, providing the 1553 interface for other subsystems and adding hardware floating point unit.

Integrating the OBC logics to the processor bus was a major challenge as the memory controller signals were completely new to us. In order to have a better understanding of the LEON3 design, it was decided to simulate the standalone LEON3 IP Core. The simulation environment for LEON3 was established using ModelSim and bash shell utilities.

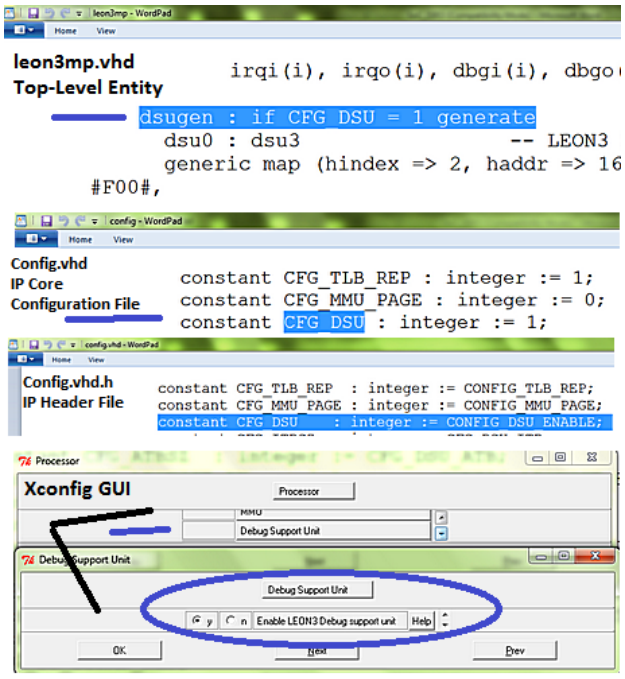

Fig. 2. Configuration, Instantiation and Interface of IP Cores

A. *Simulation of standalone LEON3 Processor*

The simulation of LEON3 was carried out by adding the necessary IP cores like Memory Controller, UART, Timer, RAM, ROM, etc and then writing simple C programs to understand and verify the processor functionality.

1) IP Core Addition

To add the necessary IP cores to the AMBA AHB Bus, the plug and play feature was utilized [12]. The IP core instantiation is done by appropriately modifying the Configuration file and instantiating the required modules and interfacing them with the processor core as shown in fig. 2.

2) IP Core Configuration Using VHDL Generics

VHDL generics have been used to configure the various IP cores required for the OBC SoC. Nearly 175 VHDL generics have been initialized to configure the necessary IP cores.

3) Application Specific Input/output Space Interface

The Memory controller of LEON3 decodes address space for PROM, RAM, DDR2 and I/O. Hence, memory mapped I/O is used. Interface logic has been designed for Memory Controller keeping in view the possible off chip provision both for memory and I/O.

Finally, the simulation was performed by writing simple test program in C, which was converted to Motorola S Record format and loaded as a PROM image in Testbench simulation model.

B. *Integrating OBC Logics*

The OBC logics provide the necessary hardware interfacing with the processor and the sensor/actuator. These logics were hooked to the processor using the Memory Controller I/O Bus. The Memory Controller provides support for 8/16/32 bit

external I/O. The read and write signals along with chip-select signals were extracted from the memory controller records. These signals were then passed through necessary combinational logic to address the OBC logic modules.

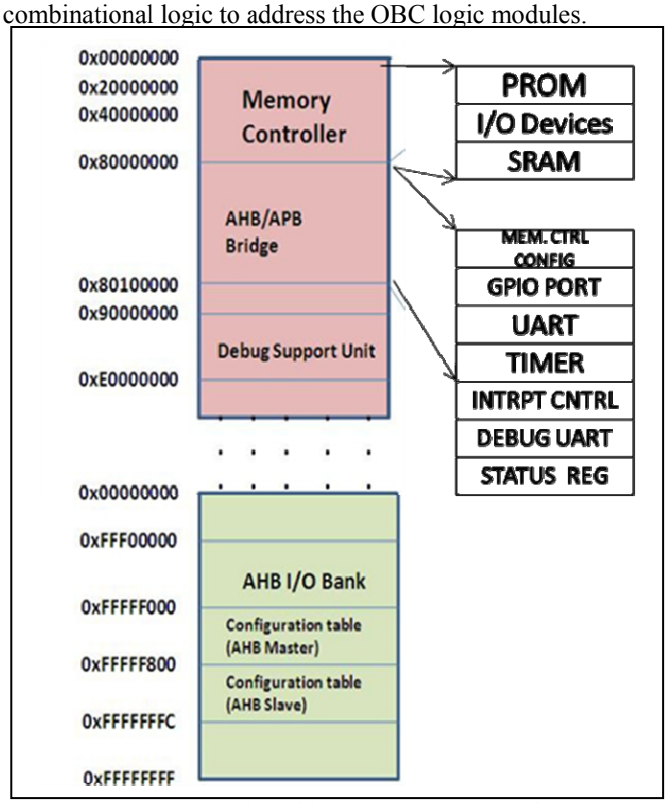

Fig. 3. Memory Map configured for interfacing LEON3 with OBC Logics

However, before adding the OBC logics to the processor, these logics were independently simulated to check for any loopholes at the desired frequency of operation. Once these logics were integrated with the LEON3 I/O Bus, Memory configuration registers were appropriately set to enable external I/O addressing and the memory mapped addresses were decoded. Finally, the simulation was done by writing and reading from the OBC specific modules using cross compiled C programs.

C. Interfacing Mil-std 1553 Bus IP Core

Actel 1553 core protocol operates at 24MHz and LEON3 can operate efficiently from 66MHz to 80MHz depending on the target FPGA. Hence there arises the problem of clock domain crossing (CDC) between the LEON3 and 1553 IP Core.

1) Problem of Clock Domain Crossing

The LEON3 clock domain has a higher frequency than the 1553 clock domain. Control signals from higher frequency LEON3 clock domain could pulse between the rising edges of a slower 1553 domain clock and not be captured into the slower 1553 clock domain. Even if the LEON3 control signals are stretched to a period greater than the 1553 clock frequency, there is a possibility of setup and hold time violations.

2) Proposed Pulse Synchronized Handshaking Clock Domain Crossing (CDC) Design Technique

Multi-cycle path (MCP) closed loop formulations techniques have been used wherein the address, data and control signals are passed asynchronously to the 1553 clock domain using a control signal This control signal generated by LEON3 is synchronized to the 1553 clock. Similarly, a READY signal is generated by the 1553 core, which is synchronized to the LEON3 clock. This signal refers to the successful completion of the transaction.

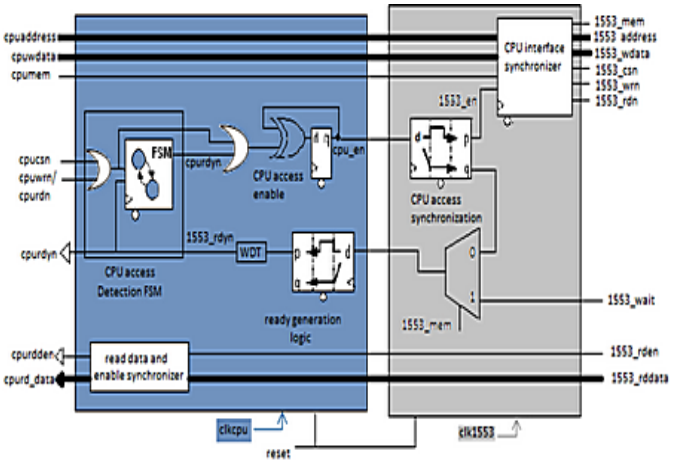

Fig. 4. Detailed block diagram of LEON3 interfaced with 1553 by CDC

3) Simulation and Verification of the LEON3-1553 Interface

Verification of the CDC handshaking module has been carried out in two stages. Firstly, handshaking module was functionally tested with 1553 IP Core. Later, the system level testing was carried out by integrating LEON3 and 1553 using the handshaking CDC module. Exhaustive test cases in simulation have been used for verification of the design. Also Static timing Analysis (STA) has been performed to eliminate any timing violations. The following test-cases comprise the subset of various test cases used for verification:

- Writing and reading into 1553 registers
- Initialization of memory , illegal command testing
- Bus-Controller (BC) message transfer, Remote Terminal (RT) to Remote Terminal transfer, BC to RT Broadcast, Mode Codes.
- BC – RT communications for all RT Transmit-Receive sub-addresses on both bus A and bus B and loop back testing.

TABLE II. LEON3 CLOCK (80 MHZ) CYCLES USED FOR DIFFERENT OPERATIONS IN DIFFERENT CONFIGURATIONS

Operation	***Standalone 1553***	***LEON3 + 1553(without CDC)***	**LEON3 + 1553 (with CDC)**
Write Register	11 cycles	12 cycles	18 cycles
Read Register	11 cycles	14 cycles	20 cycles
Write Memory	18 cycles	20 cycles	28 cycles

Operation	*Standalone 1553*	*LEON3 + 1553(without CDC)*	LEON3 + 1553 (with CDC)
Read Memory	18 cycles	22 cycles	30 cycles

D. Design and Interfacing of Hardware Floating Point Unit

Typical requirements of OBC include the need to speed up floating point operations, hence the need for a floating point unit [FPU] alongside the Integer unit. The FPU (shown in figure) is attached to the integer unit using a Floating point controller [FPC]. The designed FPU implements floating point arithmetic operations defined by the IEEE754 standard in hardware. The most common operations such as addition, subtraction and multiplication are fully pipelined. FPU division and square root operations are based on the functional iteration class of algorithms and in particular series expansion algorithms.

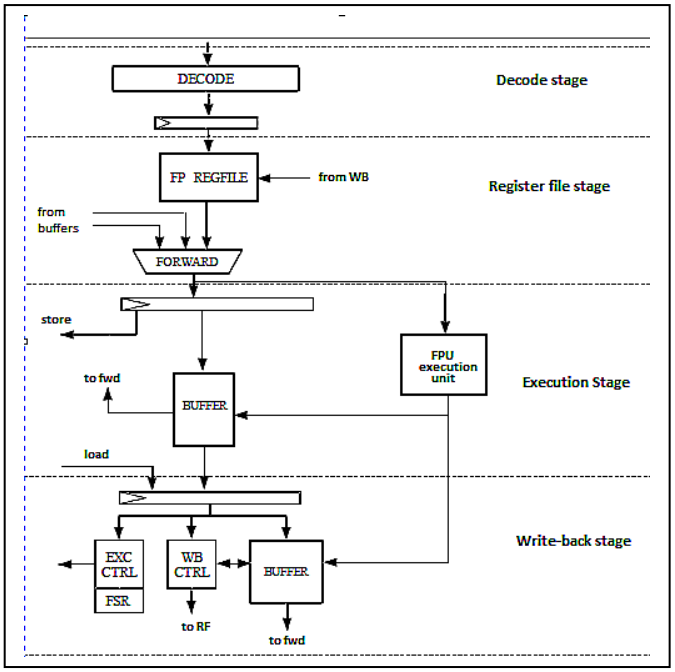

Fig. 5. Floating Point Control unit Block Diagram

1) Integration with LEON3 Pipeline

The FPU attached to a LEON processor through the FPU Control unit shown in Figure 5. The control unit receives floating point instructions from the LEON integer unit (IU) and schedules them for execution by the FPU [12]. The control unit handles the FPU register file, FPU status register and handles floating–point exceptions providing compliance with the SPARC V8 instruction scheduling.

A block diagram of floating point control unit is shown in figure. The FP instructions are decoded in the first stage of the control unit pipeline. During the second stage FP register file access is performed. Data dependencies with the result of an operation in later stage are resolved by forwarding logic.

During the next stage a FP operation is started on the FPU while the instruction is inserted in a buffer. The write back stage handles the results of the FP operations. Floating point exceptions are detected and written in the FP status registers.

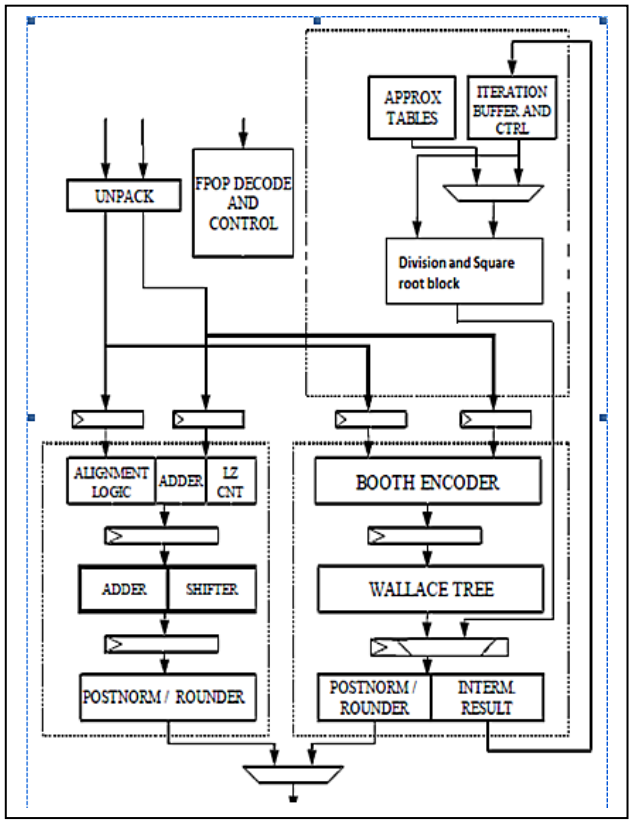

Fig. 6. FPU Block Diagram

II. HARDWARE PROTOTYPING

The Digital System on Chip (DSoC), comprising the LEON3 processor, OBC Logics, 1553 Bus IP Core, FPU has been implemented on a Xilinx Virtex 5 FPGA. Xilinx ISE 13.1 software is used to synthesize and implement the design. The Bit-stream file was loaded into the FPGA using the Xilinx Impact tool through a JTAG hardware interface. The serial UART protocol was used to load application programs. The software debugging was performed using the same RS232 connection in Eclipse IDE thereby enabling Processor-in- loop (PIL) functionality.

TABLE III. RESOURCE UTILIZATION OF XILINX VIRTEX5 FX130T FPGA

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	6,755	69,120	9%
Number used as Flip Flops	6,754		
Number used as Latch-thrus	1		
Number of Slice LUTs	15,148	69,120	21%
Number used as logic	14,865	69,120	21%
Number used as Memory	250	17,920	1%

Number of LUT Flip Flop pairs used	17079		
Number of bonded IOBs	305	640	47%
Number of BlockRAM/FIFO	35	148	23%
Total Memory used (KB)	1206	5328	22%

III. SOFTWARE DEVELOPMENT AND DEBUGGING

A Complete development environment has been set up using the GRMON debugger along with BCC cross compiler on Eclipse IDE Platform .The software design of LEON3 based SoC includes coding, compilation, linking, downloading and execution. The programs have been run in the debug mode as well as the release mode. Onboard Software modules have been ported to LEON3 and executed to compare their execution time against the performance of presently used MAR31750 processor.

A. Compilation, linking and Loading

Bare programs require a small run time kernel to run on the LEON3 processor. The run time kernel serves the following purposes:

- Initializes the Register file of Integer unit
- Initializes the Memory Controller, Timer and UART
- Copy the application into RAM
- Set the Stack pointer at the top of RAM

The program compilation requires Bare C Cross compiler [13] which is built on the lines of the GNU C Compiler [GCC. The BCC is primarily built around the New Lib C Library.

The Software development of LEON3 can be approached in two different ways. First way is running the programs in debug mode. In this mode of operation, the program is cross compiled and linked to start from base address of RAM. This means that the '.data' and '.text' segments of the cross compiled executable are directly loaded to the physical memory address by the loader. In our case, the .text segment of the program was set to start from 0x40000000 i.e. base address of RAM.

When executing applications in release mode, it is required to build a Boot PROM. This boot PROM is linked to the start-up address of the Processor on reset. In order to build a boot PROM, a boot-loader is used to initialize all peripherals and decompress the encapsulated application into RAM. This was achieved by using MKPROM utility. The Boot-loader was configured to initialize the following modules:

- RAM/ROM Size and RAM wait states
- Memory Configuration Registers
- Stack Pointer Location
- Processor Configuration Registers, etc.

Since reading and writing to the onboard flash storage device is a relatively slower process, the boot prom image was also functionally verified by executing on software simulator known as 'TSIM'. Apart from basic I/O services, TSIM also provides useful insight into the program by furnishing details such as number of instructions, the expected number of cycles, the simulation time, etc [14].

B. Real Time Debugging

GRMON Debug monitor has been used for debugging of LEON3 based SoC as in[15]-[17]. The Debug support unit acts as AHB Master in debug mode and allows access to processor registers and pipeline.

The GRMON debug monitor on the host PC connects to the Debug Support unit through a JTAG/Serial link to the target hardware. The debugging effort has been largely reduced by running the application directly from the RAM and thereby avoiding PROM reprogramming each time.

C. Porting Onboard Software Modules

Some of the presently used onboard software modules have been ported to LEON3 for measurement of execution time. These modules which were ported to the LEON3 processor are:

- Inertial Reference Unit [IRU] Processing
- Quaternion Processing
- Tacho Processing and Reaction-Wheel Spike Filter
- Three Axis Attitude Control Mode Processing
- Linear Controller

These modules, originally in Ada, were converted to C language while preserving the data types (as well as function calls) and cross compiled to be executed on the LEON3 processor. Their execution time was measured using an oscilloscope. The execution time of the software modules is listed in Table 4 for LEON3(with FPU) and MAR31750 (presently used processor in OBC). In first configuration, a Xilinx Specific Netlist has been used for FPU hardware while the second configuration does not have any FPU hardware.

TABLE IV. EXECUTION TIME (IN MICROSECONDS) OF OBC SOFTWARE MODULES FOR LEON3 VS MAR31750 PROCESSOR

Module/Configuration	***LEON3With FPU[us]***	***MAR31750 µP[us]***
IRU processing	78	810
Quaternion processing	12	803
Reaction-Wheel-Tacho processing	36	579
3-Axis Control	36	1600
Linear Controller	23	1120
Total	184	4912

IV. TESTING AND VERIFICATION

The LEON3 based DSoC has been implemented on a Xilinx Virtex 5 FPGA which is a CMOS device. CMOS devices are designed to operate in the 3.3V domain. Hence, the implemented DSoC cannot be interfaced directly to the existing onboard Test Systems installed in the laboratory because of the voltage difference between 3.3V and 5V devices. Therefore, it

was decided to test the DSoC using another 3.3V compliant FPGA.

The DSoC was tested by implementing some of the traditional Test System logics onto another Virtex5 FPGA. Here, one FPGA board acts as OBC while other is the Test System. All the DSoC OBC logics have been verified by using this methodology wherein the Test System would read the output of the DSoC and provide necessary inputs to the DSoC.

V. PERFORMANCE RESULTS

The performance of the developed SoC has been verified using OBC specific functions like Q Propagation for which the time taken with earlier processor MAR31750 is available [18]. The program has been run both with and without FPU with integer, float and double data (64 bit) types [19]. The comparison of execution time is detailed in Table 5.

A. CONFIGURATION

Processor Frequency : 80 MHz
Instruction Cache : 16 kB [2 sets of 8 kB/set]
Data Cache : 8 kB direct mapped
Hardware Multiplier : None
Floating Point Unit : GRFPU [754 Compliant]

The relatively high execution time of the Quaternion Propagation Algorithm is due to the following reasons:

- The Processor is in Debug mode and hence the Pipeline is idle.
- The Code does not utilize instruction scheduling and is non-optimized

TABLE V. QUATERNION PROPAGATION EXECUTION TIME FOR LEON3 @ 80 MHZ

Data Types	***With Floating Point Unit [us]***	***Without Floating Point Unit [us]***
Float [I Cache + D Cache]	8	278
Only I Cache	8	326
Only D cache	10	1164
No Cache	10	1251
Double[I cache + D Cache]	10	458
Only I cache	10	520
Only D Cache	12	2028
No Cache	12	2123

However, the Quaternion Algorithm propagation was also executed on the LEON3 processor running in release mode at 80 MHz. Table 6 details the results in terms of execution time.

It may be noted that this Q-Propagation algorithm was taking 0.8 ms on MAR-31750 processor (running at 12 MHz) which has been reduced to 10-12 us with LEON-3(running at 80MHz).

TABLE VI. QUATERNION PROPAGATION EXECUTION TIME FOR LEON3 @ 80 MHZ IN RELEASE MODE

Data Type: Double	***With Floating Point Unit [us]***	***Without Floating Point Unit [us]***
With Hardware Multiplier/Divider	10	204
Without Hardware Multiplier/Divider	12	368

VI. FUTURE WORK

The future work includes building fault-tolerance for harsh space environment and porting to radiation tolerant FPGAs like Actel RTAX series. ASIC design of the complete onboard computer on a single chip will require mixed signal design for integrating analog circuitry and digital modules which is being taken up parallel. The appropriate target technology for the ASIC will be selected to meet the onboard requirements with respect to radiation and temperature and the performance of the same will be established.

REFERENCES

[1] Jiri Gaisler, "LEON-1 Processor - First Evaluation Results," European Space Research and Technology Centre [ESTEC].

[2] Juan A Carrasco , Victor Sune, "An ROBDD-Based Combinatorial Method for the Evaluation of Yield of Defect-Tolerant Systems-on-Chip," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, Vol. 17, no. 2, pp. 207-220 , Feb. 2009.

[3] Sergio De Florio et al, "Performance comparison of microprocessors for Space-based navigation applications," *7th IAA Symposium of Small Satellites for Earth observations*, May 2009.

[4] Damjan Lampret, "OpenRISC 1200 IP Core Specification" Version 0.2, April 2001.

[5] *RAD750 Board Hardware User's Manual*, Document Number 234A53 3, December 20, 2000.

[6] *LEON3FT-RTAX Datasheet and User manual*, Version 1.5, June 2011, www.gaisler.com.

[7] *Mongoose-V 32-bit MIPS Microprocessor*, Architecture Description, Revision 1.3, January 1997.

[8] Luo et al, "A High-reliable SoC onboard computer using LEON3" IEEE Electronics 2012, *IEEE International Conference on Computer Science and Automation Engineering,* 2012.

[9] *GRLIB IP Core User's Manual*, Version 1.0.19, Sep. 2008, www.gaisler.com.

[10] Yahoo Groups http://tech.groups.yahoo.com/group/leon_sparc/

[11] Zhou Zhonghua et al, "SoPC Design Based on LEON3 SoC Platform" IEEE Microelectronics and Electronics 2009, *Asia Pacific Conference for Post graduate Research,* Jan 2009.

[12] Nagendra et al, "Scalable LEON3 based SoC for Multiple floating point operation" IEEE Electronics 2011, *International Conference on Current Trends in Technology,* 2011.

[13] *BCC - Bare-C Cross-Compiler User's Manual Version 1.0.*, www.gaisler.com.

[14] *TSIM2 Simulator User's Manual*, Version 2.0.18, October 2010.

[15] Daniel Hellstrom, "SnapGear Linux for LEON Version 1.37.0, November 2008.

[16] *GRMON User's manual*, Version1.1.49, April 2011, www.gaisler.com.

[17] "LEON3 Basics," Gleichmann Research University Program Tutorial.

[18] Banerjee, et al "Modern Inertial sensors and Systems", PHI Learning.

[19] Daniel Mattson et al, "Evaluation of synthesizable CPU cores," Master's thesis, Chalmers University of Technology.